9-2002

Logic Synthesis for Regular Layout using Satisfiability

Marek Perkowski
Portland State University

Alan Mishchenko
Portland State University

Citation Details

Alan Mishchenko and Marek Perkowski, "Logic Synthesis for Regular Layout using Satisfiability," Proceedings of Symposium on Boolean Problems. September, 2002, Freiberg, Germany.

Logic Synthesis for Regular Layout using Satisfiability

Marek Perkowski and Alan Mishchenko

Department of Electrical and Computer Engineering
Portland State University
Portland, OR 97207, USA
[mperkows, alanmi]@ece.pdx.edu

Abstract

In this paper, we propose a regular layout geometry called 3×3 lattice[*]*. The main difference of this geometry compared to the known 2×2 regular layout geometry is that it allows the cofactors on a level to propagate to three rather than two nodes on the lower level. This gives additional freedom to synthesize compact functional representations. We propose a SAT-based algorithm, which exploits this freedom to synthesize 3×3 lattice representations of completely specified Boolean functions. The experimental results show that the algorithm generates compact layouts in reasonable time.*

1 Introduction

Bridging together logic synthesis and layout synthesis proceeds in several directions. One approach to this problem, exemplified by [23], makes logic synthesis aware of the layout early in the design flow.

Another approach aims at creating specific layout structures with regular properties and synthesizing circuits using these structures. Research in regular structures to implement Boolean functions has started with the work of Akers [1]. The advantage of the regular layout fabrics is that they guarantee short wire length, predictable delay, and the absence of crosstalk. The disadvantage is that it may be difficult to derive compact representation for relatively complex logic functions.

Recently, regular layout fabrics are becoming popular with the new hardware implementation technologies such as single-electron transistor devices [8] and quantum dots [28]. A slide taken from [8] (Figure 4) illustrates the use of the 2x2 lattices in SET technology. There is no routing and all connections are short. Another circuits of interest for regular structure approach are called Chemically Assembled Electronic Nanotechnology (CAEN) [6][7]. CAEN is expected to offer significantly denser devices than CMOS technology. For example, a single RAM cell will require roughly 100nm^2 [6][7]. In CMOS technology, a similar cell occupies 100,000nm^2 [7].

The regular layout structure called 2x2 lattice with Shannon gates at the nodes has been first proposed in [13][14][21]. Publication [13] proposed also more general regular geometries and expansions other than Shannon, including Davio. These ideas were next expanded by several research groups [2][5][12][15][16][17][18][25] leading to the development of a number of efficient logic synthesis methods [5][9][18][19][22][26][27].

In this paper, we propose a new regular layout structure called 3x3 lattice. The 3x3 geometry preserves the close localization of logic elements characteristic of the 2x2 lattice but allows for more flexibility when implementing logic functions. We also propose the lattice synthesis algorithm, based on Boolean satisfiability, which makes use of the flexibility to reduce the number of levels and nodes in the lattice.

The rest of the paper is organized as follows. Section 2 describes the background. Section 3 presents the lattice synthesis algorithm. Section 4 gives experimental results. Section 5 concludes the paper and outlines future work.

2 Background

2.1 Expansions

Given a Boolean function F: $B^n \rightarrow B$, where $B = \{0,1\}$, the *negative (positive) cofactor* of F with respect to (w.r.t.) variable x is the Boolean function F_0 (F_1) derived by substituting into F instead of x the value 0 (1).

Let us denote F_2 the exclusive sum (EXOR) of the negative and positive cofactors: $F_2 = F_0 \oplus F_1$.

Three canonical expansions of F are defined as follows:

$F = \bar{x} F_0 \oplus x F_1$ Shannon expansion (*S*)
$F = F_0 \oplus x F_2$ Positive Davio expansion (*pD*)
$F = F_1 \oplus \bar{x} F_2$ Negative Davio expansion (*nD*)

[*] The term *lattice* is used to described the layout geometry because it is similar to a grid formed by logic gates and interconnections. The use of this term in the paper is not related to the set-theoretic concept of a lattice.

Cofactors w.r.t. two and more variables are defined as repeated cofactoring w.r.t to each variable in the set. The final result does not depend on the variable order.

For example, function $F(a,b,c) = ab \vee a\bar{b}c \vee \bar{a}bc$ has the following cofactors w.r.t. variable a:

$F_{a=0} = F(0, b, c) = bc$.

$F_{a=1} = F(1, b, c) = b \vee c$.

2.2 Lattice Types

Regular layout type discussed in this paper is called *lattices* [18]. Essentially, a lattice is a regular arrangement of gates locally interconnected to form a grid. Each gate has a control signal propagating from left to right and two data signals propagating from bottom to top. Lattice synthesis is typically performed from top to bottom when the levels of gates are synthesized one at a time until the level with constant cofactors is reached.

Differences between the 2x2 lattices and the 3x3 lattices are illustrated in Figure 1, where circles represent gates and edges represent possible interconnections. The number pairs ("2x2" and "3x3") specify the lattice geometry: the first number tells how many gates of the lower level can feed into the given gate; the second number tells how many gates of the upper level can receive the output of the given gate. The general concept of "kxk" diagrams that included 2x2 and 3x3 lattices has been presented in [13][14]. It should be noted that there exist various other 3x3 regular diagrams [2][3][13][20] that are not 3x3 lattices in the sense of this paper but have been also called "lattices" by us. However, in this paper the terms 2x2 and 3x3 lattices will refer only to structures shown in Figure 1.

Depending on the lattice type, three types of gates can be used in the nodes: Shannnon gates (MUXes), Positive Davio gates, and Negative Davio gates, created according to the three canonical expansions. Shannon lattices are built using only Shannnon gates, Kronecker lattices can have any of the three gates but only one gate type on each level. Pseudo-Kronecker lattices can have any of the three gates assigned to any node.

The lattices considered in this paper have the following additional flexibilities: (1) the data inputs of a gate can be complemented; (2) both data inputs of a gate can be connected to the same gate below. In the synthesis methods developed for 3x3 lattices, the gates are limited to only Shannon gates.

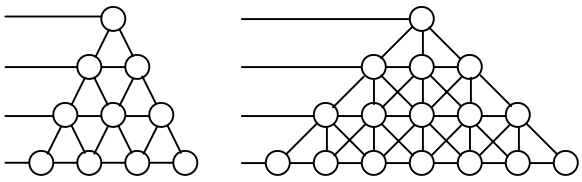

Figure 1. 2x2 and 3x3 lattices.

2.3 Comparison of 2×2 and 3×3 Geometries

Note that although, in the 3x3 lattice, a gate can receive inputs from any of the three gates on the lower level (left, center, right), no more than two gates actually provide the inputs (because each gate has only two data inputs). The synthesis algorithm can use this additional freedom for choosing two gates out of three candidates on the lower level to achieve a compact layout, with less logic levels and fewer gates.

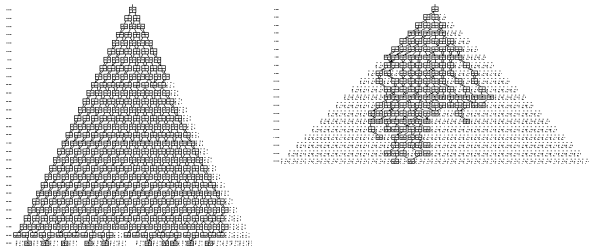

Figure 2. Layout comparison of 2x2 and 3x3 lattice for a random function of 7 variables.

Figure 2 shows the 2x2 lattice and 3x3 lattices synthesized for a randomly generated 7 variable function. The 2x2 lattice was synthesized using the tools from [25]. The 3x3 lattice was created by a new tool, which implements the algorithm presented in this paper. Both lattices are synthesized using only Shannon gates. The 2x2 lattice has 29 levels and 392 nodes. The 3x3 lattice has 20 levels and 150 nodes.

Random functions are among those that are the most difficult to implement in regular structures. Figure 2 clearly demonstrates the advantage of the 3x3 geometry. For other functions, 3x3 lattices are never larger than 2x2 lattices but the difference is less pronounced.

3 SAT-Based Lattice Synthesis

This section shows how the problem of synthesizing one level of the 3x3 lattice with Shannon gates can be reduced to a SAT instance. We assume the reader's familiarity with the basics of Boolean satisfiability as presented, for example, in [10].

3.1 Outline of the Algorithm

Synthesis of the 3x3 lattice is performed level by level. On each level, we solve the SAT problem, create the layout of that level, and proceed to the next level. Synthesis terminates when a level is reached on which all cofactors are constants.

To formulate the SAT problem for one level of the lattice, we consider all the nodes on the level following immediately after the given one. The given level is called

the *upper level*; the level following immediately after the given one is called the *lower level*.

We formulate the set of requirements representing all possible routings of cofactors from the upper level to the lower level. Only non-constant cofactors are considered for routing. The constraints generated for all the nodes of the lower level are added to the set of all constraints representing the SAT instance.

3.2 Variables

Let us consider node N on the lower level. In the 3x3 lattice, there are six cofactors (c_1,…,c_6) that can potentially be routed to this node. These cofactors come from the nodes on the left (L), in the center (C), and on the right (R) nodes, with respect to the node N (Figure 1).

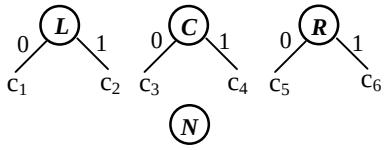

Figure 3. The routing choices for node N.

A group of SAT variables (x^N_1, …, x^N_k) is associated with each node N of the lower level. These variables represent mutually exclusive possibilities of joining in node N the cofactors coming from the upper level.

Suppose the cofactors (c_1,…,c_6) are different non-constant Boolean functions. The group of six variables (x^N_1, …, x^N_6) is used to represent the possibilities that node N receives only one cofactor. Another group of six variables (x^N_7, …, x^N_{12}) is used to represent the possibility that N receives a pair of cofactors combined using join-vertex operation [13][18].

The join-vertex operation is defined for cofactors coming from different nodes and having opposite polarity. This is why there are only six pairs, namely (c_1,c_4), (c_2,c_3), (c_3,c_6), (c_4,c_5), (c_1,c_6), and (c_2,c_5).

3.3 Clauses

The clauses of the SAT problem are divided into two categories: *covering constraints* and *closure constraints*. The covering constraints specify the requirement that each non-constant cofactor on the upper level is routed to the lower level. The closure constraints represent two types of requirements: (1) each cofactor on the upper level is routed no more than once, and (2) each node on the lower level is used no more than once.

There are as many covering constraints as there are non-constant cofactors on the upper level. Each covering constraint is the disjunction of variables, which represent routing choices involving the given cofactor. Suppose, for some cofactor, there are m such variables. The mutual exclusiveness of these variables translates into $m(m-1)/2$ closure constraints of type (1), specifying that no two variables are equal to 1 at the same time.

The closure constraints of type (2) are similar. For each node N_i on the lower level, they represent the mutual exclusiveness of variables in (x^{Ni}_1, …, x^{Ni}_{ki}). There are $k_i(k_i-1)/2$ such constraints for node N_i with k_i associated variables. Because $k \le 12$, the number of these constraints does not exceed 66 for any node N_i.

3.4 The Number of Variables and Clauses

In this subsection, we approximate the number of clauses and constraints in a SAT problem, which arises in the lattice synthesis on level n.

In the 3x3 lattice, level n (the upper level) consists of $2n$ - 1 nodes. The level n+1 (the lower level) consists of $2n$ + 1 nodes. Each node on the lower level can have up to 12 variables, so the upper bound V on the number of variables is

$$V(n) = 12(2n+1).$$

The number of covering constraints is equal to the number of cofactors on level n, which is $2(2n-1)$. In the worst case, the number of the closure constraints of type (1) and type (2) is $66(2n-1)$ and $66(2n+1)$, respectively. This yields the upper bound C on the number of clauses in the SAT problem

$$C(n) = 2n - 1 + 66(2n-1) + 66(2n+1) \approx 266n.$$

The worst-case number of variables and clauses grows *linearly* with the number of lattice levels. For example, for n=10, there are at the most V = 252 variables and C = 2660 clauses. When some of the cofactors on the upper level are constants or equal up to complementation, it is possible to introduce less than 12 variables for nodes of the lower level. These special cases can be used to significantly reduce the routing choices. As a result, the actual number of clauses is typically two times smaller than the worst-case estimation.

3.5 Weighted SAT Problem

Each variable (x^{Ni}_1, …, x^{Ni}_{ki}) in the SAT problem is assigned a cost. The negative assignment of a variable has the cost 0. The positive assignment of a variable representing the join-vertex operation has a positive cost which is higher compared to the cost of the variable representing routing of the cofactors or joining of the cofactors equal up to complementation.

In solving the SAT problem, we look for satisfying variable assignments having the smallest cost. Such solutions would guarantee that there are relatively few nodes with the join-vertex operation, which helps reducing the number of levels in the lattice.

The complexity of the weighted SAT problem is NP-complete.

3.6 Trading Quality for Runtime

To simplify the complexity of the SAT problem, each level can be split into parts, for which SAT is solved independently. There is no routing of cofactors across the boundaries of the parts.

This approach to reduce complexity of SAT allows for trading the layout quality for runtime. The smaller are the individual SAT problems, the faster they are solved. On the other hand, splitting a level into many parts leads to the sub-optimal layout near the boundaries.

In practice, we found that having the parts composed of 5-7 nodes allows for a reasonable tradeoff between the optimality of the solution and the runtime of weighted SAT problems, which in this case takes approximately 0.1 seconds per instance.

4 Experimental Results

We have implemented the 3x3 lattice synthesis in a C program using the BDD package CUDD [24] for the manipulation of Boolean functions and the MINCOV package in SIS [22] to iteratively solve the weighted Boolean satisfiability problem.

We tested our program on selected MCNC benchmarks. The resulting lattices were written into BLIF files and verified against the initial specification of the benchmark functions. The runtime for any particular example was dominated by the runtime of SAT solver and did not exceed three seconds on a 933MHz Pentium III PC under MS Windows 2000.

Table 1 shows the comparison of our results with those published in [4] for 2x2 lattices. The 2x2 lattices are called Pseudo-Symmetric Binary Decision Diagrams (PSBDDs). They use only Shannon gates and allow for the same additional flexibilities: (1) the data inputs of a gate can be complemented; (2) both data inputs of a gate can be connected to the same gate below.

Column “Name” gives the name of the benchmark circuit. Column “Outs” gives the total number of outputs in the circuit and, in parentheses, the particular output used for testing. Similarly, column “Ins” gives the total number of input in the circuit and, in parentheses, the number of variables in the support of the output used for testing. Column “2x2 Lattice” gives the number of levels and nodes in the lattice reported in [4]. Column “3x3 Lattice” gives the number of levels and nodes in our implementation.

The experimental result in Table 1 show that synthesizing benchmark functions into the 3x3 lattices helps reducing the number of levels by 24% and the number of nodes by 56%, which speaks for the efficiency of the SAT-based synthesis algorithm.

Table 1. Experimental results.

Name	**Outs**	**Ins**	**2 x 2 Lattice**		**3 x 3 Lattice**	
			levels	**nodes**	**levels**	**nodes**
apex7	37(30)	49(17)	25	148	19	33
clip	5(1)	9(9)	18	103	9	30
	5(2)	9(9)	27	220	9	30
cm162a	5(3)	14(10)	11	24	10	18
cps	109(1)	24(22)	26	134	24	61
	109(2)	24(18)	26	164	21	66
	109(3)	24(22)	39	342	30	128
duke2	29(3)	22(15)	18	52	15	69
	29(6)	22(17)	18	47	17	36
	29(18)	22(15)	22	92	15	44
example2	66(23)	85(16)	21	52	16	45
	66(59)	85(14)	17	31	14	21
	66(63)	85(13)	15	37	13	26
frg2	139(99)	143(20)	22	189	20	28
	139(100)	143(19)	28	164	20	61
sao2	4(2)	10(10)	18	71	14	56
	4(3)	10(10)	16	73	14	65
	4(4)	10(10)	16	68	11	46
Total			383	2011	291	863
Ratio, %			100	100	**76**	**43**

5 Conclusions

We presented a regular layout structure called 3x3 lattice. We demonstrated that this structure gives additional freedom to implement Boolean functions. We proposed a synthesis algorithm, which uses this freedom to generate the lattice layout with a smaller number of levels and nodes, compared to the known 2x2 lattice synthesis algorithms.

The proposed synthesis algorithm reduces the problem of lattice synthesis on a level to an instance of weighted Boolean satisfiability. We showed that the worst-case number of variables and clauses in the SAT instances is linear in the level number to be synthesized. An additional advantage of the SAT formulation is that it allows for an efficient trade-off between the layout quality and runtime.

The future work in this area may include generalizing the synthesis algorithm to synthesis Kronecker and Pseudo-Kronecker 3x3 lattices and integrating our tools with one of the powerful SAT solvers developed recently, for example [11]. It is important to note that these ideas can be expanded to regular structures based on multiple-valued and fuzzy logic expansions and their reversible counterparts [2][3][13][15][16][17][20][21] and are thus applicable to many emerging nano-technologies for computing, especially such as Single Electron Transistors and Quantum Dots.

References

[1] S. B. Akers. A Rectangular Logic Array, *IEEE Trans. on Computers*. Vol. C-21, pp. 848-857, 1972

[2] Al-Rabadi and M. Perkowski, Shannon and Davio Sets of New Lattice Structures for Logic Synthesis in Three-Dimensional Space, *Proc. RM'01*

[3] Al-Rabadi and M. Perkowski, New Classes of Multi-valued Reversible Decompositions for Three-Dimensional Layout, *Proc. RM'01*.

[4] M. Chrzanowska-Jeske, Y. Xu, M. Perkowski, Logic Synthesis for a Regular Layout. *VLSI Design: An International Journal of Custom-Chip Design, Simulation, and Testing*, 1999.

[5] B.T. Drucker, C.M. Files, M. A. Perkowski, and M.Chrzanowska-Jeske. Polarized Pseudo-Kronecker Symmetry with an Application to the Synthesis of Lattice Decision Diagrams. *Proc. ICCIMA'98 Conference*, pp. 745-755.

[6] P. Farm and E. Dubrova, Technology Mapping for Chemically Assembled Electronic Nanotechnology, *Proc. IWLS '02.*

[7] S. Goldstein and M. Budiu, Nanofabrics: Spatial computing using molecular electronics, *Proc. 28th Annual International Symposium on Computer Architecture*, (Gothenborg, Sweden), June 2001.

[8] H. Hasegawa, A. Ito, Ch. Jiang, and T. Muranaka, Atomic Assisted Selective MBE Growth of InGaAs Linear and Hexagonal Nanowire Networks For Novel Quantum Circuits, *Proc. 4th Intern Workshop on Novel Index Surfaces (NIS'01)*, Sept 16-20, Apset France.

[9] P. Lindgren, R. Drechsler, B. Becker, Synthesis of Pseudo-Kronecker Lattice Diagrams. *Proc. of Intl. Worshop of Applications of Reed-Muller Expantions to Circuit Synthesis, 1999*, Victoria, B. C., Canada, pp. 197 - 204.

[10] J. P. Marques-Silva, K. A. Sakallah, GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Comp*. Vol. 48, No. 5, May 1999, pp. 506-521.

[11] M. W. Moskewicz et al., Chaff: Engineering an Efficient SAT Solver. *Proc. DAC'01*, pp. 530-535.

[12] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, S. I. Long, Wave Steering in YADDs: A Novel Non-Iterative Synthesis and Layout Technique. *Proc. DAC'99*, pp. 466-471.

[13] M. Perkowski, and E. Pierzchala, New Canonical Forms for Four-Valued Logic, *Report, Electrical Engineering Department, PSU* . 1993.

[14] E. Pierzchala, and M. Perkowski, *Patent #5,959,871*, September 28, 1999. *Programmable Analog Array Circuit.*

[15] M. Perkowski, E. Pierzchala, and R. Drechsler, Ternary and Quaternary Lattice Diagrams for Linearly-Independent Logic, Multiple-Valued Logic and Analog Synthesis, *Proc. ISIC-97*, Singapur, 10-12 Sept.1997.

[16] M. Perkowski, L. Jozwiak, R. Drechsler, and B. J. Falkowski**,** Ordered and Shared, Linearly Independent, Variable-Pair Decision Diagrams, *Proc. First International Conference on Information, Communications and Signal Processing*, ICICS'97, Singapur, 9-12 Sept. 1997. Session 1C1: Spectral Techniques and Decision Diagrams.

[17] M. Perkowski, E. Pierzchala, and R. Drechsler, Layout-Driven Synthesis for Submicron Technology: Mapping Expansions to Regular Lattices, *Proc. First International Conference on Information, Communications and Signal Processing, ICICS'97*, Singapur, 9-12 Sept. 1997. Session 1C1: Spectral Techniques and Decision Diagrams.

[18] M. Perkowski, M. Chrzanowska-Jeske, and Y. Xu, Lattice Diagrams Using Reed-Muller Logic. *Proc. of Intl. Worshop of Applications of Reed-Muller Expansions to Circuit Synthesis, 1997*, Oxford Univ., U.K., pp. 85 - 102.

[19] M. A. Perkowski, M. Chrzanowska-Jeske, and Yang Xu, Multi-Level Programmable Arrays for Sub-Micron Technology based on Symmetries, *Proc. ICCIMA'98 Conference*, pp. 707-720, February 1998, Australia, published by *World Scientific.*

[20] M. Perkowski, A. Al-Rabadi, P. Kerntopf, A. Mishchenko and M. Chrzanowska-Jeske, Three-Dimensional Realization of Multiple-Valued Functions using Reversible Logic, Invited Talk, *Proc. Workshop on Post-Binary Ultra-Large Scale Integration Systems (ULSI),* , pp. 47 - 53, May 21, 2001, Warsaw, Poland.

[21] E. Pierzchala, M. A. Perkowski, S. Grygiel, A Field Programmable Analog Arrray for Continuous, Fuzzy and Multi-Valued Logic Applications, *Proc. ISMVL'94*, pp. 148 - 155, Boston, MA, May 25-27, 1994.

[22] E. Sentovich, et al., SIS: A System for Sequential Circuit Synthesis, *Tech. Rep. UCB/ERI, M92/41*, ERL, Dept. of EECS, Univ. of California, Berkeley, 1992.

[23] A. Singh, G. Parthasarathy, M. Marek-Sadowska, Interconnect Resource-Aware Placement for Hierarchical FPGAs. *Proc. ICCAD '01*, pp. 132-137.

[24] F. Somenzi. *CUDD Package, Release 2.3.1*. http://vlsi.Colorado.EDU/~fabio/CUDD/cuddIntro.html

[25] VLSI Design Automation Laboratory. *Pseudo-Symmetric Kronecker Functional Decision Diagrams.* http://web.pdx.edu/~suresh/pskfdd/main.php

[26] W. Wang, M. Chrzanowska-Jeske, Optimizing Pseudo-Symmetric Binary Decision Diagrams Using Multiple Symmetries. *Proc. IWLS'98*, pp. 134-140.

[27] W. Wang, M. Chrzanowska-Jeske, Generating Linear Arrays Using Symmetry Chain. *Proc. IWLS'99*, pp.115 -119.

[28] T. Yamada, Y. Kinoshita, S. Kasai, H. Hasegawa, Y. Amemiya, Quantum Dot Logic Circuits Based on Shared Binary-Decision Diagram, *Jpn. J. Appl. Phys.* Vol. 40, 2002, pp. 4485-4488, Part1, No. 7, July 2001.

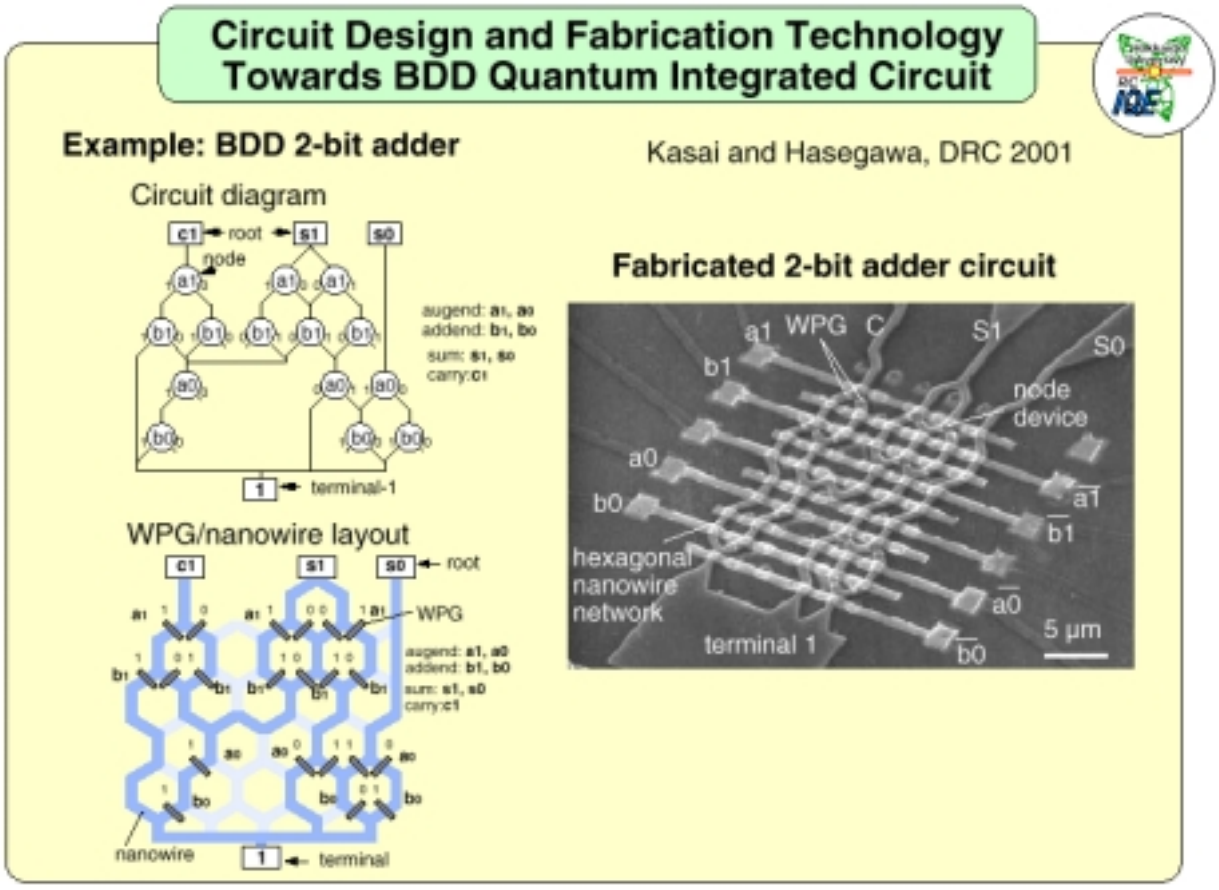

Figure 4. Explanation of using lattice structure (called hexagonal structure) based on Single Electron Transistors to realize a 2-bit adder [8].